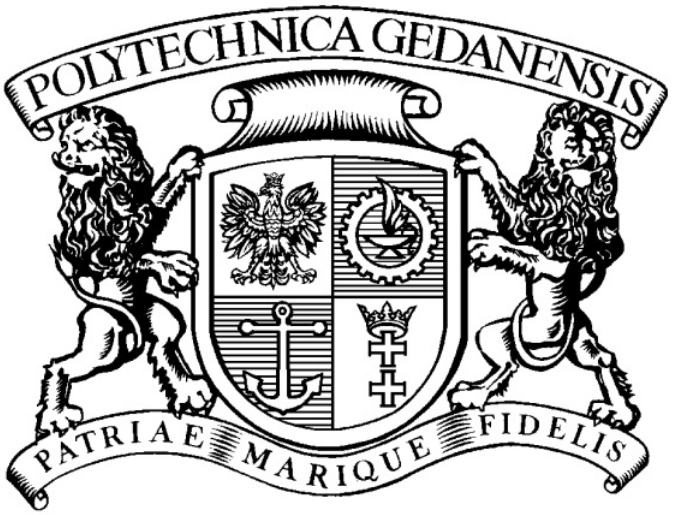


Politechnika Gdańska

Wydział Elektroniki, Telekomunikacji i Informatyki

Projekt grupowy
Wizualizacja grafów za pomocą biblioteki Prefuse

Autorzy:

	nr indeksu
Anna Jaworska	106306
Radosław Kleczkowski	106317
Piotr Kunowski	106345
Piotr Orłowski	106386

17 stycznia 2010

Spis treści

1 Zlecenie projektowe

Symbol projektu: 3@KASK	Opiekun projektu: mgr inż. Tomasz Boiński
Nazwa Projektu: Wizualizacja grafów za pomocą biblioteki Prefuse	

Nazwa Dokumentu: Zlecenie projektowe	Nr wersji: 0.2
Odpowiedzialny za dokument: Piotr Kunowski	Data pierwszego sporządzenia: 30 marca 2009
Przeznaczenie: Wewnętrzne	Data ostatniej aktualizacji: 17 stycznia 2010

Historia dokumentu

Wersja	**Opis modyfikacji**	**Rozdział/strona**	**Autor modyfikacji**	**Data**
1	Stworzenie dokumentu	wszystkie	Grupa projektowa	30.03.09
2	Dodanie forumułki o RUP	4	Anna Jaworska	15.04.09

1.1 Cele i opis projektu

Celem projektu jest utworzenie biblioteki umożliwiającej wizualizację ontologii zapisanych w OWL API. Do tego celu należy wykorzystać język Java oraz bibliotekę Prefuse. Szczególny nacisk w projekcie należy położyć na:

- Wizualizację elementów niejawnych (np. klasy anonimowe wyrażone poprzez unie, przecięcie itp. oraz dziedziczenie po tych klasach, łączenie wielu odwzorowań niejawnych)
- Wizualizację powiązań między klasami oraz innymi elementami grafu
- Udokumentowanie stworzonej biblioteki za pomoca JavaDoc
- Zapewnienie możliowości integracji uzyskanej biblioteki z istniejącą aplikacją OCS

1.2 Zleceniodawca

mgr inż. Tomasz Boiński, Katedra Architektury Systemów Komputerowych, Wydział Elektroniki, Telekomunikacji i Informatyki, Politechnika Gdańska.

1.3 Zleceniobiorca

Studenci wydziału Elektorniki, Telekomunikacji i Informatyki, Katedry Architektury Systemów Komputerowych.

Imię i nazwisko	**Rola**	**E-mail**	**Telefon**
Piotr Kunowski	Kierownik projektu	p.kunos@gmail.com	781-765-187
Anna Jaworska	Członek zespołu	valanthe86@gmail.com	666-089-481
Radosław Kleczkowski	Członek zespołu	radoslaw1201@gmail.com	brak
Piotr Orłowski	Członek zespołu	cmsptcp@gmail.com	brak

1.4 Zakres prac

Pierwszy etap projektu

1. Studium wykonalności - stworzenie następujących dokumentów:
   - Zlecenie projektowe
   - Harmonogram
   - Słownik
   - Studium wykonalności
2. Analiza wymagań - stworzenie następujących dokumentów:
   - Specyfikacja wymagań
   - Specyfikacja przypadków użycia
3. Analiza obiektowa - stworzenie następujących dokumentów:
   - Model klas
   - Model dynamiki
   - Specyfikacja przypadków testowych
4. Prototyp - stworzenie kodu i dokumentów:
   - Prototyp klas
   - Opis prototypu
5. Odbiór projektu - stworzenie następujących dokumentów:
   - Plakat
   - Prezentacja

Drugi etap projektu

1. Iteracja 1
    - Aktualizacja dokumentacji
    - Implementacja
    - Testowanie
2. Iteracja 2
    - Aktualizacja dokumentacji
    - Implementacja
    - Testowanie
3. Podsumownie
    - Aktualizacja dokumentacji
    - Podsumowanie

2 Infrastruktura projektu

Symbol projektu: 3@KASK	Opiekun projektu: mgr inż. Tomasz Boiński
Nazwa Projektu: Wizualizacja grafów za pomocą biblioteki Prefuse	

Nazwa Dokumentu: Infrastruktura projektu	Nr wersji: 1.0
Odpowiedzialny za dokument: Anna Jaworska	Data pierwszego sporządzenia: 31.03.09
Przeznaczenie: WEWNĘTRZNE	Data ostatniej aktualizacji: 20.04.09

Historia dokumentu

Wersja	**Opis modyfikacji**	**Rozdział/strona**	**Autor modyfikacji**	**Data**
0.0	Stworzenie	wszystkie	Anna Jaworska	31.03.09
1.0	Wpisanie używanych narzędzi	wszystkie	Anna Jaworska	20.04.09

2.1 Organizacja zespołu projektu

Nazwa roli	Osoba(y)
Kierownik projektu	Piotr Kunowski
Specjalista ds. testów	Radosław Kleczkowski
Analityk ds. ontologii	Piotr Orłowski
Analityk ds. Prefuse	Piotr Kunowski
Analityk główny	Anna Jaworska
Programiści	cały zespół

2.2 Dokumentacja

Dokumenty tworzone sa na podstawie następujących szablonów składownych na SVN:

- szablon.tex
- notatka_szablon.tex

2.3 Narzędzia i wymiana informacji

2.3.1 Narzędzia programistyczne

- Netbeans 6.5

2.3.2 Biblioteki i środowisko

- JAVA ver 6
- Prefuse ver prefuse-beta20071021
- OWL API ver 2.1.1

2.3.3 Komunikacja w zespole

- Gadu-gadu
- Email
- Telefonicznie
- Wymiana dokumentacji przez SVN, materiałów dodatkowych przez email

2.3.4 Tworzenie dokumentacji

- Dokumenty w LateX
- na SVN wrzucamy pliki tex i ich wersje pdf

2.3.5 Inne używane programy

Rysowanie notacji dla ontologii Inkspace i Dia

UML Netbeans

Ontologie Programy używane jako wzorcowe zarówno w kwestii wizualizacji jak i implementacji: Protege, GrOWL.

Harmonogramy GanttProject

3 Studium wykonalności

Symbol projektu: 3@KASK	Opiekun projektu: mgr inż. Tomasz Boiński
Nazwa Projektu: Wizualizacja grafów za pomocą biblioteki Prefuse	

Nazwa Dokumentu: Studium wykonalności	Nr wersji: 0.8
Odpowiedzialny za dokument: Anna Jaworska	Data pierwszego sporządzenia: 31.03.09
Przeznaczenie: WEWNĘTRZNE	Data ostatniej aktualizacji: 16.06.09

Historia dokumentu

Wersja	**Opis modyfikacji**	**Rozdział/strona**	**Autor modyfikacji**	**Data**
0.0	Przygotowanie zarysu dokumentu i określenie zakresu badań	wszystkie	Anna Jaworska	31.03.09
0.1	Zdefiniowanie wymagań	3	Cały zespół	31.03.09
0.2	Dołaczenie opisu poprawnego tworzenia bibliotek	3.5	Radosław Kleczkowski	01.04.09
0.3	Dołączenie opisów bibliotek graficznych	6.2	Piotr Kunowski	02.04.09
0.4	Opis uwarunkowań prawnych i rozszerzenie opisu wariantów	5, 6.1, 7	Anna Jaworska	06.04.09
0.5	Uzupełnienie braków	wszystkie	Cały zespół	07.04.09
0.6	Dołączenie opisu odmian języka OWL i korekta	6.1, 7	Piotr Orłowski	07.04.09
0.7	Korekta	6.1, 7	Radosław Kleczkowski i Piotr Orłowski	15.06.09
0.8	Dołączenie harmonogramów	8, 9	Radosław Kleczkowski	16.06.09

3.1 Założenia realizacji studium

3.1.1 Podstawa wykonania i temat studium

Studium wykonywane jest przede wszystkim aby określić możliwe sposoby realizacji projektu. Ma także za zadanie zebranie i podsumowanie informacji potrzebnych zespołowi do realizacji projektu.

3.1.2 Cel studium

Celem studium jest zbadanie na potrzeby projektu *Wizualizacja grafów za pomocą biblioteki Prefuse*:

- jak należy tworzyć biblioteki w technologii Java
- jakich mechnizmów wizualizacji grafów dostarczają biblioteki Java
- czy realizacja projektu za pomocą Prefuse jest odpowiednim rozwiązaniem
- jaki standard OWL powinien być wspierany przez wytworzony produkt

3.1.3 Ograniczenia

Do podstawowych ograniczeń należą:

- konieczność realizacji projektu w języku Java
- konieczność wykorzystania wersji bibliotek zgodnych z użytymi w OCS
- limit czasowy projektu

3.2 Stan istniejący

3.2.1 Inne systemy i zasoby mające wpływ lub będące pod wypływem planowanego produktu

- OCS - Ontology Creation System
- OWL API ver 2.1.1 - API do przetwarzania plików w formacie OWL zgodnych ze standardem W3C; ta wersja API została użyta w projekcie OCS
- biblioteki graficzne - w szczególności Prefuse

3.2.2 Istniejące na rynku podobne rozwiązania

- Protege - bardzo znany system do edycji i wizualizacji ontologii autorstwa Stanford University. Napisany w języku Java. Ze względu na fakt, iż jest aplikacją standalone, wykorzystującą stosunkowo duże zasoby systemowe i trudną do integracji z portalem OCS, nie może zostać wykorzystana jako gotowe rozwiązanie.

3.2.3 Problem i motywacja wdrożenia nowego produktu

Nowa biblioteka powinna powstać aby:

- ułatwić programistom wizualizację ontologii
- zapewnić API pozwalające na bezpośrednią translację OWL na postać graficzną
- zapewnić rozwiązane przenośnie i uniwersalne

3.3 Ogólne wymagania stawiane produktowi i ich priorytety

Wymienione wymagania mają charakter orientacyjny, pozwalający nakreślić zakres problemu jaki ma pokrywać projekt. Szczegółową definicję wymagań będzie zawierać dokument *Specyfikacji wymagań*. W szczególność możliwe jest, że niektóre z wymienionych poniżej wymagań zostaną usunięte lub zmienione oraz to, że mogą zostać dodane inne wymagania.

3.3.1 Użytkownicy

Użytkownikami biblioteki będą programiści tworzący aplikacje wizualizujące ontologie. Inicjalnie będą to programiści związani z projektem OCS, później mogą to być dowolni inni programiści chętni do korzystania z biblioteki.

3.3.2 Dane

Obsługiwane formaty Biblioteka powinna obsługiwać te same formaty danych co OWL API (zgodne ze specyfikacją W3C):

- RDF
- RDF Schema
- OWL Lite
- OWL DL
- OWL Full

Wczytywanie danych Ponadto dane te powinny być przekazywane poprzez obiekt OWL API.

Modyfikowalność danych Biblioteka powinna udostępniać metody do modyfikacji wczytanych danych i możliwość zapisania zmienionych danych. Dane powinny być dostarczane użytkownikowi w postaci obietków OWL API. Biblioteka nie musi sprawdzać czy zmiany wprowadzone przez użytkownika są logicznie poprawne.

3.3.3 Funkcjonalność

Zakładamy, że biblioteka będzie zawierać następujące funkcjonalności:

- wizualizacja elmentów OWL
- definiowanie przez użytkownika własnych akcji dla zdarzeń okna (np. klinięcie, przeciągnięcie wierzchołka grafu)
- standardowe definicje zdarzeń okna
- wczytywanie, modyfikowanie i zapis ontologi
- definiowanie parametrów wyglądu, w szczególności ilości widocznych poziomów grafu

3.3.4 Wymogi techniczno - technologiczne: Standard tworzenia biblioteki

Nie istnieją żadne formalne zalecenia dotyczące tworzenia bibliotek JAVA. Są jednak pewne zalecenia co do stosowanych praktyk [1]:

1. **Odpowiednie kapsułkowanie.** Publiczne powinny być jedynie te klasy i metody, które są istotne dla użytkownika i z których będzie on bezpośrednio korzystał.
2. **Możliwość debugowania.** Użytkownik powinien mieć możliwość debugowania kodu biblioteki, bez konieczności znajomości każdego jej szczegółu.
3. **Przejrzystość.** Kod biblioteki powinien być odpowiednio udokumentowany za pomocą javadoc. W szczególności, bardzo dokładnie należy opisać klasy oraz metody publiczne.
4. **Łatwość użycia.** Biblioteka powinna zawierać klasy, pokazujące przykłady wykorzystania jej klas i metod.

[1] Greg Travis. Build your own java library. publikacja `http://www.digilife.be/quickreferences/PT/BuildyourownJavalibrary.pdf`.

5. **Rozszerzalność.** Struktura wewnętrzna biblioteki powinna być odpowiednio podzielona na klasy (wykorzystując klasy abstrakcyjne i interfejsy. Dzięki temu użytkownik będzie miał możliwość stworzenia własnych klas, rozszerzających funkcjonalność biblioteki.

6. **Uniwersalność.** Biblioteka powinna mieć jasno określony problem, który rozwiązuje. Wyniki powinny być podane użytkownikowi w wygodny dla niego sposób (lub na kilka sposobów), który będzie umożliwiał wykorzystanie biblioteki w różnych aplikacjach. Innymi słowy, biblioteka powinna udostępniać łatwy i przejrzysty dla użytkownika interfejs.

7. Biblioteka powinna być napisana w taki sposób, aby użytkownik spojrzał na nią i mógł powiedzieć: "Wow, to jest dokładnie to, czego potrzebuję i dokładnie tak samo bym to napisał!".

3.4 Ogólna ocena ryzyka i planowany sposób zarządzania nim

Schemat opisu czynnika ryzyka

ID czynnika	RISKXX
Nazwa czynnika	Nazwa
Opis czynnika	Opis...
Sposób zarządzania	Opis..

3.4.1 Czynniki ryzyka

ID czynnika	RISK01
Nazwa czynnika	Problemy logistyczne zespołu
Opis czynnika	Uwzględniamy możliwość wystąpienia problemów osobistych członków zespołu powodujących ich wyłączenie z prac.
Sposób zarządzania	Jeśli ktoś zostanie wyłączony z prac, reszta zespołu musi podzielić między siebie jego obowiązki i informować osobę wyłączoną o postępach, tak aby ona miała wgląd w postęp prac, które miała wykonywać i kontynuować je po niedyspozycji.

ID czynnika	RISK02
Nazwa czynnika	Problemy członków zespołu na uczelni
Opis czynnika	Możliwe jest powstanie zaległości związanych z innymi uczelnianymi obowiązkami
Sposób zarządzania	Członek zespołu musi zgłosić swoje problemy reszcie zespołu. W zależności od sytuacji termin wykonania jego zadań zostanie przedłużony lub zadania te przejmie ktoś inny.

ID czynnika	RISK03
Nazwa czynnika	Niedostępność opiekuna/klienta
Opis czynnika	Z różnych przyczyn niezależnych od zespołu opiekun może stać się niedostępny.
Sposób zarządzania	Wszelkie problemy wymagające, według zespołu, poznania opinii opiekuna będą musiały zostać rozwiązanie poprzez podjęcie decyzji przez zespół bez wsparcia. Wszelkie problemy organizacyjne związane z projektem grupowym powinny być pod nieobecność opiekuna zgłaszane do katedralnego koordynatora projektów grupowych.

ID czynnika	RISK04
Nazwa czynnika	Niewystarczająca wiedza programisty
Opis czynnika	W trakcie pisania kodu może okazać się, że programista z powodu nieznajomości bibliotek/metod/praktyk zacznie mieć problemy z wydajnym kodowaniem (zacznie popełniać częste błędy, pracować bardzo wolno).
Sposób zarządzania	Osoba mająca problemy z danym kodem powinna zgłosić to reszcie zespołu. Jeśli ograniczenia czasowe na to pozwolą dostanie ona dodatkowy czas na wykonanie zadania. Jeśli nie będzie to możliwe, zadanie zostanie przekazanie osobie będącej w stanie poradzić sobie z zagadnieniem lub zostanie podzielone między większą liczbę osób.

ID czynnika	RISK05
Nazwa czynnika	Awaria SVN
Opis czynnika	Serwer SVN nie jest dostępny lub działa w sposób nieporządany.
Sposób zarządzania	Problem należy niezwłocznie zgłosić opiekunowi i oczekiwac na jego interwencję.

3.5 Uwarunkowania prawne i inne

Docelowy produkt będzie własnością Katedry Architektury Systemów Komputerowych wydziału Elektroniki, Telekomunikacji i Informatyki Politechniki Gdańskiej. Należy zadbać o to aby używane w projekcie biblioteki były na licencjach pozwalających na użycie w produkcie zamkniętym.

3.6 Proponowane rozwiązania

Proponowane rozwiązania zostaną rozważone pod względem wersji OWL oraz biblioteki graficznej.

3.6.1 Wersja OWL

Lite
- zawiera bazowe elementy OWL i RDF
  - typy: Class, Property, Individual
  - podstawowe nierówności, zależności, charakterystyki
  - elementarna kardynalność
  - adnotacje
- pozwala budować hierarchię elementów
- wymaga separacji typów

DL
- zawiera wszystkie elementy języka OWL Lite
- dodatkowo zawiera zaawansowane elementy języka OWL
  - ma rozwiniętą obsługe zależności między elementami podstawowymi
  - obsługuje kardynalność w jej pełnej formie
- można go bezpośrednio mapować na logikę opisową SHOIN - jest rozstrzygalny
- tą wersję obsługuje portalSubsystem

DL
- zawiera wszystkie elementy OWL DL
- nie wymaga separacji typów
- ma mniejsze ograniczenia od OWL DL
- nie ma w nim gwarancji rozstrzygalności dla wnioskowań

Full

Należy zwrócić uwagę, że specyfikacja OWL jest dobrze zdefiniowana (rekomendacja W3C[2]) co sprawia, że zachodzi spójność pomiędzy jej elementami. Zaimplementowanie wersji bardziej rozwiniętej oznacza, że wymogi dla wersji niższej także zostaną spełnione.

[2]Frank van Harmelen Deborah L. McGuinness. Owl web ontology language overview. publikacja elektroniczna, luty 2004.`http://www.w3.org/TR/2004/REC-owl-features-20040210/`

3.6.2 Proponowane biblioteki do wizualizacji grafów

Prefuse jest elastycznym pakietem dostarczającym programiście narzędzia do przechowywania danych, manipulowania nimi oraz ich interaktywnej wizualizacji. Biblioteka jest rozwijana w całości w języku Java. Może być wykorzystana do budowania niezależnych aplikacji, wizualnych komponentów rozbudowanych aplikacji oraz tworzenia apletów.

Podstawowe cechy i elementy:

- kilkadziesiąt algorytmów i metod wizualizacji danych m.in: ForceDirectedLayout, RadialTreeLayout, NodeLinkTreeLayout, SquarifiedTreeMapLayout
- dynamiczne rozmieszczanie i animacje
- transformacje, przekształcenia geometryczne oraz przybliżanie/oddalanie obrazu
- podstawowym elementem struktury danych jest krotka
- krotki mogą być tworzone bezpośrednio w aplikacji lub na podstawie zewnętrznych danych
- wbudowany język zapytań do filtrowania danych
- tworzenie struktur danych na podstawie zewnętrznych plików (CSV, XML) oraz bazy danych
- klasy wspomagające synchronizację danych pomiędzy tabelami Prefuse a bazą danych
- Prefuse posiada licencję BSD

Piccolo jest zastawem narzędzi używanych przy tworzeniu graficznych aplikacji. Często wykorzystywana do tworzenie interfejsów użytkownika. w których elementy są przybliżane i oddalane. Istnieją trzy wersje tej biblioteki: Piccolo.Java, Piccolo.NET oraz PocketPiccolo.NET. Posiada Licencje BSD.

JUNG (Java Universal Network/Graph Framework) Biblioteka przeznaczona do wizualizacji danych za pomocą grafów oraz sieci. Umożliwia wizualizację nie tylko grafów prostych, ale m.in. multigrafów, digrafów oraz grafów posiadających wagi i etykiety na wierzchołkach i krawędziach. Biblioteka posiada podstawowe algorytmy grafowe. Została napisana w całości w Javie i wydana na licencji BSD.

JGraph Napisana w pełni w Javie biblioteka do wizualizacji grafów kompatybilna ze Swingiem. Posiada wiele ciekawych opcji wizualizacji zarówno wierzchołków jak i krawędzi grafów. Poza algorytmami wizualizacji w jej skład wchodzą podstawowe algorytmy grafowe. Została wydana na licencji LGPL.

3.7 Rekomendowany wariant

OWL: Po zapoznaniu się ze specyfikacją stworzoną przez W3C najbardziej sensownym wydaje się być wykorzystanie wersji DL języka OWL. Dodatkowo wersja ta była dotychczas wykorzystywana przez portalSubsystem. Grupa nie odrzuca możliwości zaimplementowania obsługi wersji OWL Full, która pod względem zawartych w niej elementów zasadniczo nie różni się od wersji DL. Na jej niekorzyść przemawia jednak argument w postaci tego, że umożliwia pewne niejasności w prezentacji (szczególnie pod względem rozróżniania typów).

Biblioteka: Po uważnym przejrzeniu bibliotek najbardziej użyteczne wydają się Prefuse oraz Piccolo. Ze względu na dostępność dużej ilości przykładowego kodu wykorzystującego Prefuse w portalSubsystem wykorzystana zostanie biblioteka Prefuse. Ponadto opinie wyrażone w pracy magisterskiej Andrzeja Jakowskiego silnie przemawiają na korzyść Prefuse.

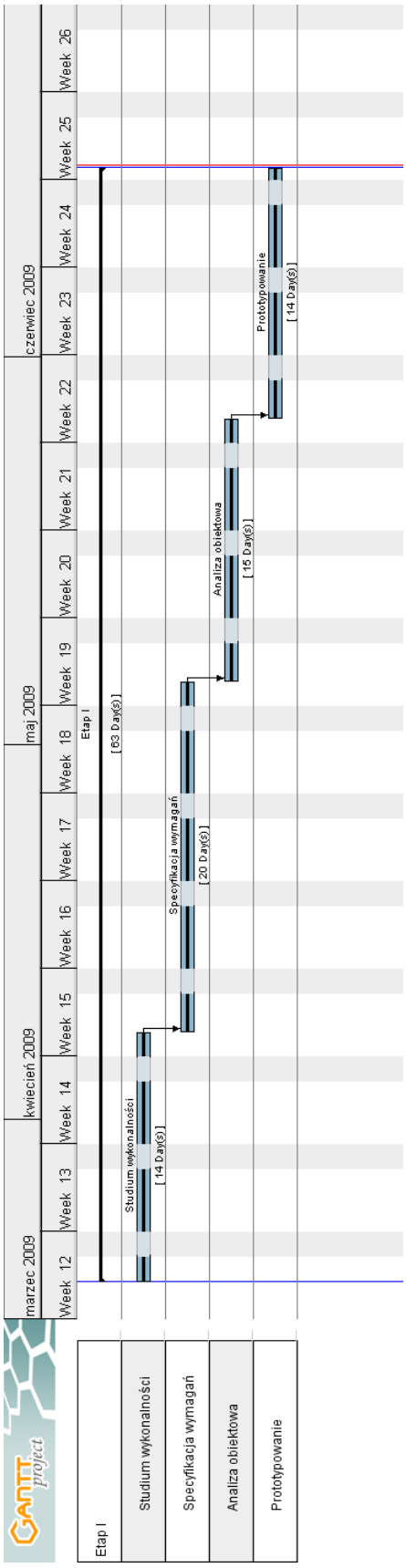
3.8 Strategia i wstępny harmonogram

Ze względu na doświadczenie zespołu z Rational Unified Process (trzej członkowie zespołu uprzednio zrealizowali projekt w tej metodyce), zostanie on zastosowany z uwzględnieniem stosowanych dla charakteru projektu modyfikacji, w szczególności:

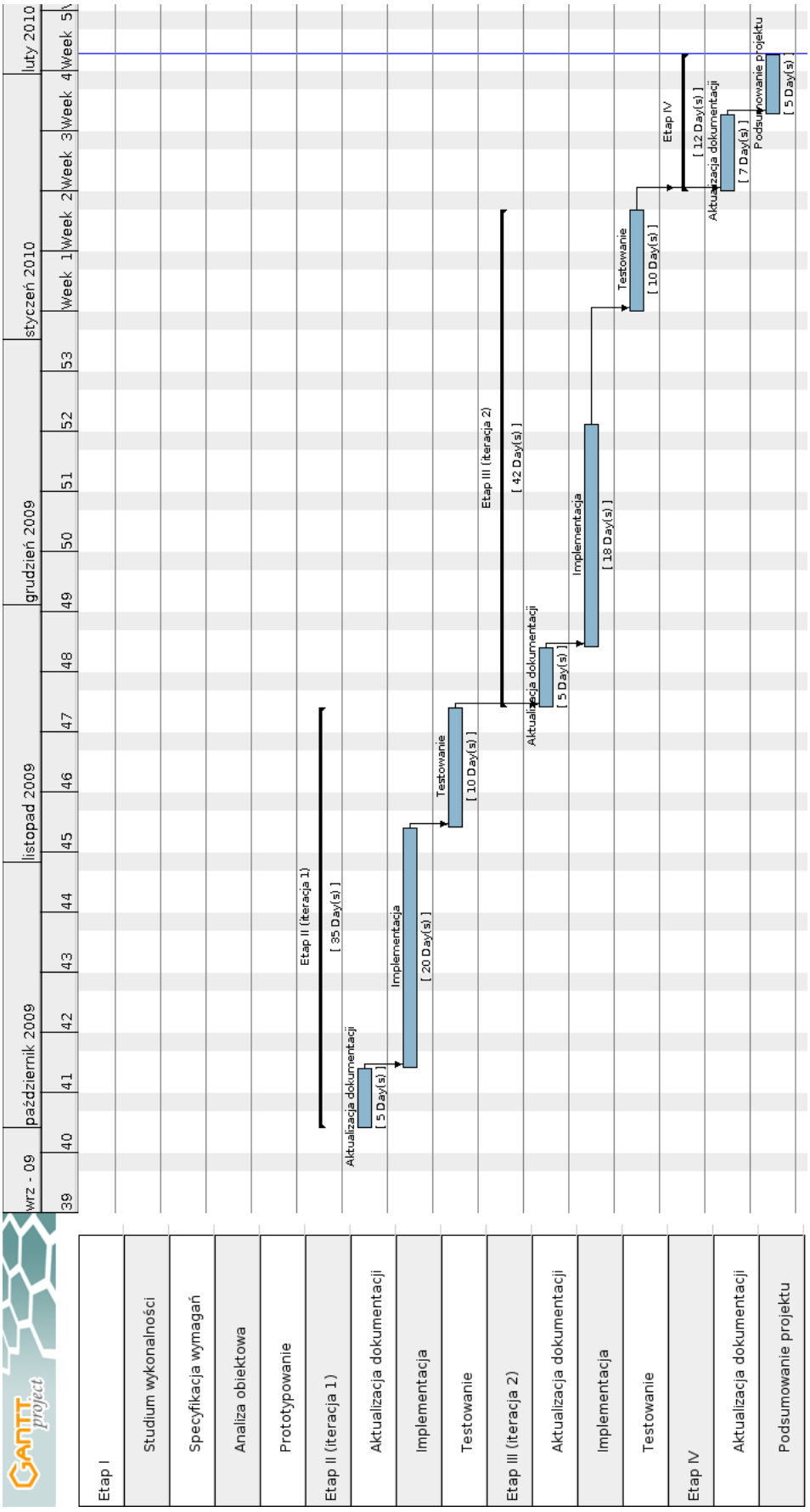
- celem projektu jest wytworzenie biblioteki, więc nie pojawią się typowe diagramy warstwy danych
- model interfejsu graficznego zostanie zastąpiony modelem interfejsów/funkcjonalności zewnętrznych udostępnianych przez pakiety i/lub klasy
- modele dynamiki zostaną okrojone do ilości faktycznie potrzebnej programistom

Pomimo ustalenia harmonogramu z terminami oddania dokumentów należy wziąć pod uwagę charakter metodyki RUP, która zakłada przyrostowe wytwarzanie dokumentacji - w późniejszych etapach projektu pojawią się zmodyfikowane wersje wytorzonych wcześniej dokumentów.

3.8.1 Harmonogram na I semestr



3.8.2 Harmonogram na II semestr



4 Specyfikacja wymagań systemowych

Symbol projektu: 3@KASK	Opiekun projektu: mgr inż. Tomasz Boiński
Nazwa Projektu: Wizualizacja grafów za pomocą biblioteki Prefuse	

Nazwa Dokumentu: Specyfikacja wymagań systemowych	Nr wersji: 0.7
Odpowiedzialny za dokument: Piotr Orłowski	Data pierwszego sporządzenia: 15 kwietnia 2009
Przeznaczenie: DLA KLIENTA	Data ostatniej aktualizacji: 17 stycznia 2010

Historia dokumentu

Wersja	**Opis modyfikacji**	**Rozdział/strona**	**Autor modyfikacji**	**Data**
1	Stworzenie	wszystkie	Grupa projektowa	15.04.09
2	Wpisanie celów i wymogów ogólnych	cele	Grupa projektowa	16.04.09
3	Wpisanie funkcjonalnosci wizualizacyjnych		Grupa projektowa	28.04.09
4	Opis wymagań		Grupa projektowa	05.05.09
5	Zmiana kolorów Property (SomeValuesFrom i AllValuesFrom)	Projekt wizualizacji	Grupa projektowa	18.05.09
6	WJ001 - klasa Thing w grafie	Wymagania jakościowe	Grupa projektowa	25.05.09
7	Korekta	Całość	Piotr Kunowski	16.06.09

4.1 Cele systemu

4.1.1 Cele biznesowe

CB001	Ułatwienie pracy programistom tworzącym aplikacje wizualizujące ontologie
Opis:	Istnieje zapotrzebowanie na bibliotekę tłumaczącą OWL bezpośrednio na elementy graficzne.
Źródło:	Wstępna specyfikacja projektu
Priorytet:	bardzo ważne

CB002	Ułatwienie zakończenia projektu OCS
Opis:	Moduł wizualizujący ontolgie w OCS wymaga modernizacji i rozbudowy funkcjonalności. Zapewnienie biblioteki wizualizującej ontologie ułatwi i przyspieszy ten proces.
Źródło:	Klient - mgr inż. Tomasz Boiński
Priorytet:	bardzo ważne

CB003	Zwiększenie aktrakcyjności portalu OCS
Opis:	Poprawa estetyki modułu wizualizującego ontologię moze przyczynic się do sukcesu portalu po jego wdrożeniu.
Źródło:	Klient - mgr inż. Tomasz Boiński
Priorytet:	mało ważne

4.1.2 Cele funkcjonalne

CF001	Intuicyjne API
Opis:	API powinno być uznane za intuicyjne w opinii członków zespołu i klienta.
Źródło:	Klient - mgr inż. Tomasz Boiński
Priorytet:	średnio ważne

CF002	Dobra dokumentacja
Opis:	Przygotowanie dokumentacji w Javadoc ułatwi pracę użytkownikom biblioteki.
Źródło:	Klient - mgr inż. Tomasz Boiński
Priorytet:	bardzo ważne

CF003	Wizualizacja ontologii
Opis:	Stworzenie biblioteki, która pozwoli na wizualizacje obiektów OWL API przy użyciu odpowiedniej biblioteki graficznej.
Źródło:	Specyfikacja projektu
Priorytet:	bardzo ważne

CF004	Umożliwienie graficznej edycji i dodawania obiektów OWL API
Opis:	Dostarczenie tej funkcjonalności ułatwi tworzenie programów z interfejsem pozwalającym na edycję ontologii zapisanych w OWL API.
Źródło:	Klient - mgr inż. Tomasz Boiński
Priorytet:	średnio ważne

CF005	Udostępnienie informacji do debuggowania
Opis:	Biblioteka powinna wysyłać komunikaty informacyjne, ostrzegawcze oraz informujace o błędach na strumień udostępniony użytkownikowi.
Źródło:	Standard tworzenia biblioteki
Priorytet:	średnio ważne

4.2 Otoczenie systemu

4.2.1 Użytkownicy

Specyfika projektu nie definiuje użytkowników systemu.

4.2.2 Systemy zewnętrzne

Specyfika systemu nie wymaga definiowaia systemów zewnętrznych.

4.3 Przewidywane komponenty systemu

4.3.1 Podsystemy

Specyfika projektu sprawia, że podsystemy nie będa rozpatrywane.

4.3.2 Komponenty sprzętowe

Specyfika projektu sprawia, że komponenty sprzętowe nie będa rozpatrywane.

4.3.3 Programowe

KS001	Prefuse
Opis:	Biblioteka graficzna do wizualizacji grafów w języku Java
Powiązania:	
Źródło:	Specyfikacja projektu
Priorytet:	bardzo ważne

KS002	OWL API
Opis:	Biblioteka do przetwarzania ontologii zapisanych w języku OWL. Napisana w języku Java.
Powiązania:	
Źródło:	Specyfikacja projektu
Priorytet:	bardzo ważne

4.4 Wymagania funkcjonalne

WF001	Udostępnienie kilku algorytmów wizualizacji
Opis:	Biblioteka powinna udostępniać kilka trybów prezentacji grafów (np. w formie drzewa, w formie gwiazdy i innych).
Dotyczy:	CF003
Źródło:	klient - mgr Tomasz Boiński
Powiązania:	WF002
Priorytet:	średnio ważny

WF002	Parametryzacja trybów wizualizacyjnych
Opis:	Domyślne parametry w trybach wizualizacji (takie jak długość krawędzi grafu, automatyczne układanie) powinny zostać dobrane w taki sposób, by obraz był przejrzysty, stabilny i czytelny.
Dotyczy:	CF003
Źródło:	klient - mgr Tomasz Boiński
Powiązania:	WF001
Priorytet:	średnio ważny

WF003	Udostępnienie strumienia błędów
Opis:	Biblioteka będzie udostępniać strumień danych, w którym znajdą się komunikaty o błędach. Strumień ten będzie mógł zostać wykorzystany przez użytkownika.
Dotyczy:	CF005
Źródło:	klient - mgr inż. Tomasz Boiński
Powiązania:	
Priorytet:	ważne

WF010	Dodatkowe informacje
Opis:	Biblioteka będzie dostarczać informacje o wersji ontologii zapisane w pliku OWL oraz dodatkowe informacje o klasach (annotationProperty).
Dotyczy:	CF003
Źródło:	klient - mgr inż. Tomasz Boiński
Powiązania:	
Priorytet:	średnio ważne

4.4.1 Wymagania wizualizacji ontologii

WF004	Rozróżnialność podstawowych symboli
Opis:	Class, Individual, Property powinny mieć rozróżnialne symbole
Dotyczy:	CF003
Źródło:	klient - mgr inż. Tomasz Boiński
Powiązania:	
Priorytet:	bardzo ważne

WF005	Rozróżnialność szczególnych typów Class
Opis:	Klasa anonimowa, datatype, Thing i Nothing powinny być łatwo rozpoznawalne.
Dotyczy:	CF003
Źródło:	klient - mgr inż. Tomasz Boiński
Powiązania:	WF004
Priorytet:	ważne

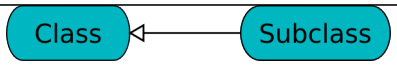
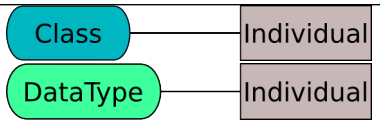
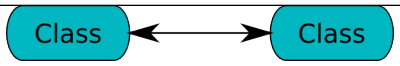
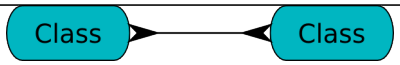
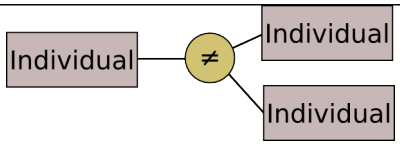
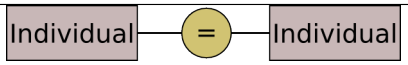
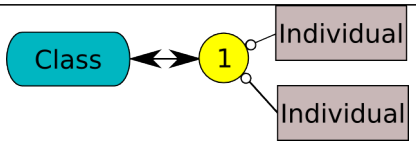
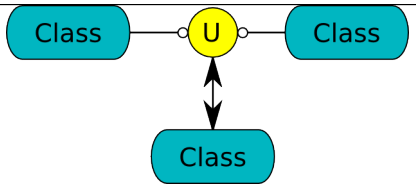
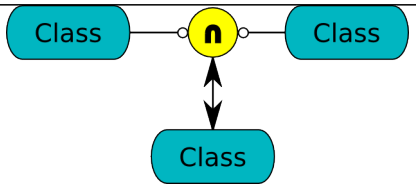
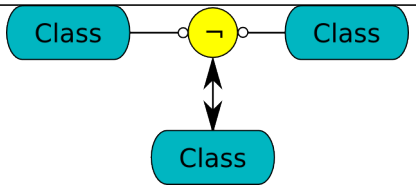
WF006	Rozróżnialność związków między klasami (Class), instancjami (Individual) oraz predykatami (Property)
Opis:	Rózne symobole dla equivalentClass, disjointWith, subClassOf, sameAs, differentFrom, allDifferent, oneOf, unionOf, intersectionOf, complementOf, subProperty, equivalentProperty, hasProperty.
Dotyczy:	CF003
Źródło:	klient - mgr inż. Tomasz Boiński
Powiązania:	WF005, WF004
Priorytet:	ważne

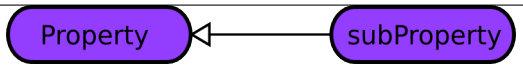
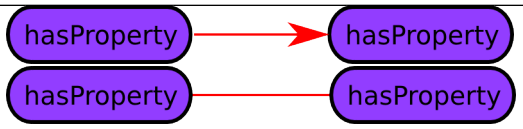
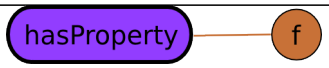
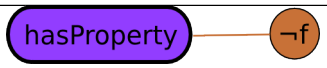
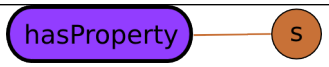
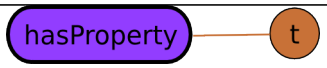
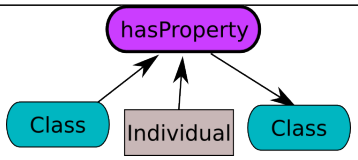
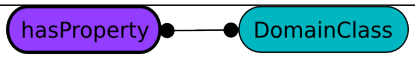
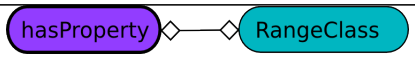
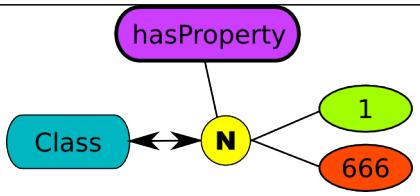
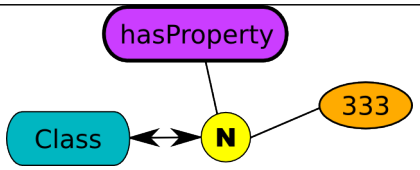
WF007	Rozróżnialność ograniczeń predykatów (Restrictions)
Opis:	Wyróżnić kardynalność (cardinality), domeny (domains) predykatów, inverseOf, właściwości predykatów (transitive, symmetric, functional, inverseFunctional).
Dotyczy:	CF003
Źródło:	klient - mgr inż. Tomasz Boiński
Powiązania:	WF004
Priorytet:	ważne

WF008	Podświetlanie wybranych związków i powiazań.
Opis:	Podświetlać subklasy danej klasy po ich wybraniu myszką po zdefiniowanym zdarzeniu; podobnie subproperty i complex class.
Dotyczy:	CF003
Źródło:	klient - mgr inż. Tomasz Boiński
Powiązania:	WF006
Priorytet:	mało ważne

WF009	Możliwość definiowania zdarzeń.
Opis:	Użytkownik będzie mógł pod uchwyty zdarzeń podpinać własne funkcje obsługi.
Dotyczy:	CF003, CF004
Źródło:	klient - mgr inż. Tomasz Boiński
Powiązania:	
Priorytet:	mało ważne

4.4.2 Projekt wizualizacji

Identyfikator:	Nazwa	Wizualizacja
PW001:	Thing	
PW002:	Nothing	
PW003:	Class	
PW004:	Individual	
PW005:	Property	
PW006:	Datatype	
PW007:	Anonymous Class	
PW008:	Subclass	
PW009:	instanceOf	
PW010:	equivalentClass	
PW011:	disjointWith	
PW012:	differentFrom / allDifferent	
PW013:	sameAs	
PW014:	oneOf	
PW015:	unionOf	
PW016:	intersectionOf	
PW017:	complementOf	

PW018:	subProperty	
PW019:	inverseOf (property)	
PW020:	equivalentProperty	
PW021:	functionalProperty	
PW022:	inverseFunctionalProperty	
PW023:	symmetricProperty	
PW024:	transitiveProperty	
PW025:	hasProperty	
PW026:	domain	
PW027:	range	
PW028:	allValuesFrom	
PW029:	someValuesFrom	
PW030:	minCardinality / maxCardinality	
PW031:	cardinality	

4.5 Wymagania na dane

WD001	Obsługa obiektów OWL API
Opis:	Biblioteka będzie przystosowana do pobierania, obróbki i zwracania obiektów OWL API.
Powiązania:	
Źródło:	Klient - mgr inż. Tomasz Boiński
Priorytet:	bardzo ważne

4.6 Wymagania jakościowe

4.6.1 Wymagania w zakresie wiarygodności

WJ001	Poprawność wizualizacji
Opis:	Wszystkie wizualizowane elementy powinny pochodzić z ontologii otrzymanej na wejściu programu. Program nie powinien dodawać własnych elementów (np. wywnioskowanych). Wyjątkowo dla klas, które nie mają zdefioniowany nadklas zostanie utworzony związek z klasą Thing.
Powiązania:	WJ002
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	bardzo ważne

WJ002	Kompletność wizualizacji
Opis:	Jeżeli biblioteka nie wizualizuje danej funkcji OWL API informacja o tym powinna znaleźć się w strumieniu błędów.
Powiązania:	CF005, WJ001, WD001
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	ważne

4.6.2 Wymagania w zakresie wydajności

Brak wymogów wydajnościowych ze względu na specyfikę projektu.

4.6.3 Wymagania w zakresie elastyczności

WJ003	Obsługiwane wersje Javy
Opis:	Biblioteka powinna wspierać wersje Javy 1.5 i nowsze.
Powiązania:	
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	bardzo ważne

WJ004	Obsługiwane wersje OWL API
Opis:	Powinna istnieć możliwość podpięcia zewnętrznego OWL API (wybranego przez użytkownika/programistę).
Powiązania:	
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	bardzo ważne

4.6.4 Wymagania w zakresie użyteczności

Ze względu na przyjętą metodykę wytwarzania oprogramowania zagadnienie to zostanie rozpatrzone w przyszłości.

4.7 Sytuacje wyjątkowe

Ze względu specyfikę projektu sytuacje wyjątkowe nie będą rozpatrywane.

4.8 Dodatkowe wymagania

4.8.1 Wymagania sprzętowe

Ze względu na specyfikę projektu wymagania sprzętowe nie będą rozpatrywane.

4.8.2 Wymagania programowe

WD003	JVM
Opis:	Do skorzystania z biblioteki niezbędna jest JVM.
Dotyczy:	CF001, CF002
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	ważne

4.8.3 Inne wymagania

WI001	Dokumentacja w javadoc
Opis:	Wszystkie ważne klasy i funkcje powinny mieć odpowiednią dokumentację w formacie javadoc.
Dotyczy:	CF001, CF002
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	ważne

WI002	Dokumentacja w języku angielskim
Opis:	Dokumentacja wszystkich funkcji i klas powinna posiadać angielską wersję językową.
Dotyczy:	CF001, CF002
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	mało ważne

WI003	Dokumentacja w języku polskim
Opis:	Dokumentacja wszystkich funkcji i klas powinna posiadać polską wersję językową.
Dotyczy:	CF001, CF002
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	ważne

WI004	Nazwy zmiennych i funkcji w języku angielskim
Opis:	Nazwy zmiennych i funkcji powinny zostać dobrane w języku angielskim i zgodnie ze standardami programowania w javie
Dotyczy:	CF001, CF002
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	ważne

4.9 Kryteria akceptacyjne

KA001	Spełnione są podstawowe wymagania wymienione w dokumencie SWS
Opis:	Spełnione są wszystkie wymagania ważne i bardzo ważne zdefiniowane w SWS.
Dotyczy:	wszystkie wymagania ważne i bardzo ważne
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	ważne

KA002	Biblioteka współpracuje z OWL API dostarczonym przez KASK
Opis:	Biblioteka współpracuje z OWL API dostarczonym przez KASK zbudowanym na podstawie OWL API ver 2.1.1
Dotyczy:	WJ004
Źródło:	klient - mgr inż. Tomasz Boiński
Priorytet:	ważne

5 Analiza obiektowa

Symbol projektu: 3@KASK	Opiekun projektu: mgr inż. Tomasz Boiński
Nazwa Projektu: Wizualizacja grafów za pomocą biblioteki Prefuse	

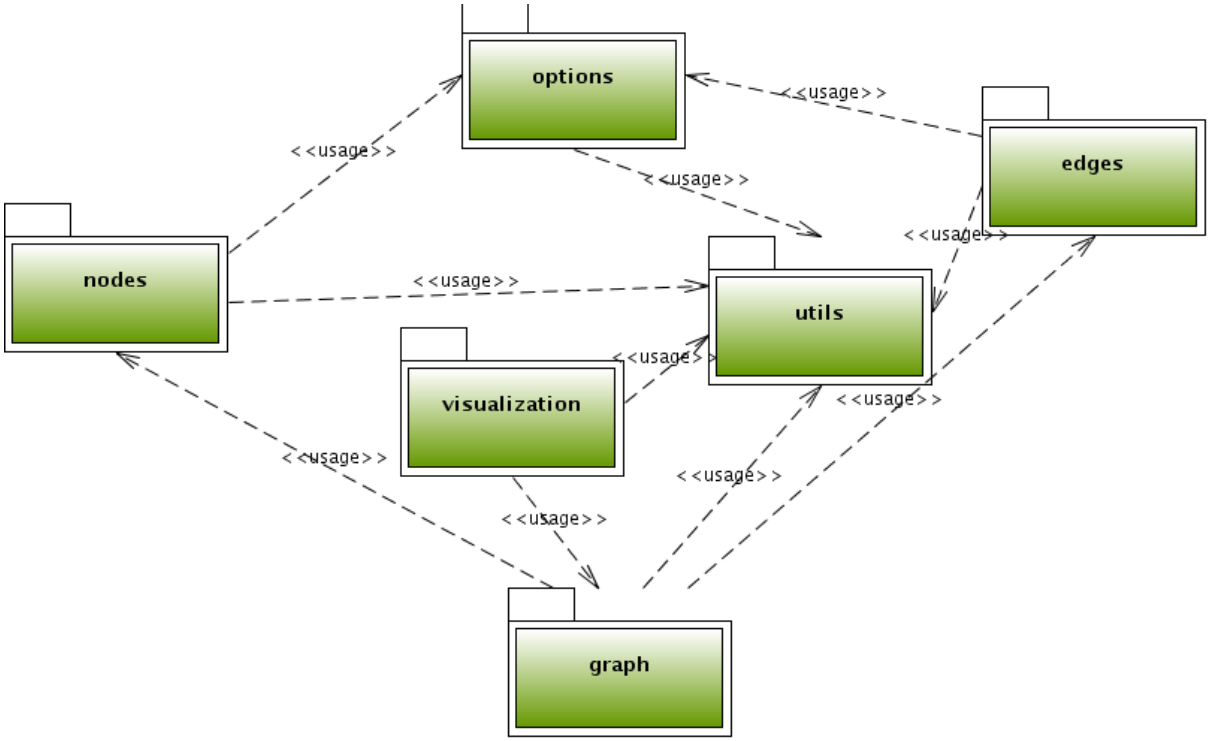
Nazwa Dokumentu: Analiza obiektowa	Nr wersji: 2.0
Odpowiedzialny za dokument: Piotr Kunowski	Data pierwszego sporządzenia: 23 maja 2009
Przeznaczenie: DLA KLIENTA	Data ostatniej aktualizacji: 17 stycznia 2010

Historia dokumentu

Wersja	Opis modyfikacji	Rozdział/strona	Autor modyfikacji	Data
1	Stworzenie	wszystkie	Grupa projektowa	23.05.09
1.1	Dodano pakiet Utils	1, 3	Anna Jaworska	2.06.09
2	Dodano zaktualizowane diagramy oraz opisy klas	wszystkie	Grupa projektowa	16.06.09

5.1 Pakiety

5.1.1 Diagram



5.1.2 Opis pakietów

P001	options
Opis:	Pakiet zawierający klasy z polami opisującymi różne (modyfikowalne) ustawienia wizualizacji takie jak: kolory, grubość linii itp.
Interfejsy:	
Realizowane wymagania:	WF002, WF001, WI004
Priorytet:	średnio ważne

P002	nodes
Opis:	Pakiet z klasami odpowiedzialnymi za wizualizację i przechowywanie danych o wierzchołkach.
Interfejsy:	
Realizowane wymagania:	WF004, WF005, WF006, WF007, WI004
Priorytet:	bardzo ważne

P003	edges
Opis:	Pakiet z klasami odpowiedzialnymi za wizualizację i przechowywanie danych o krawędziach.
Interfejsy:	
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	bardzo ważne

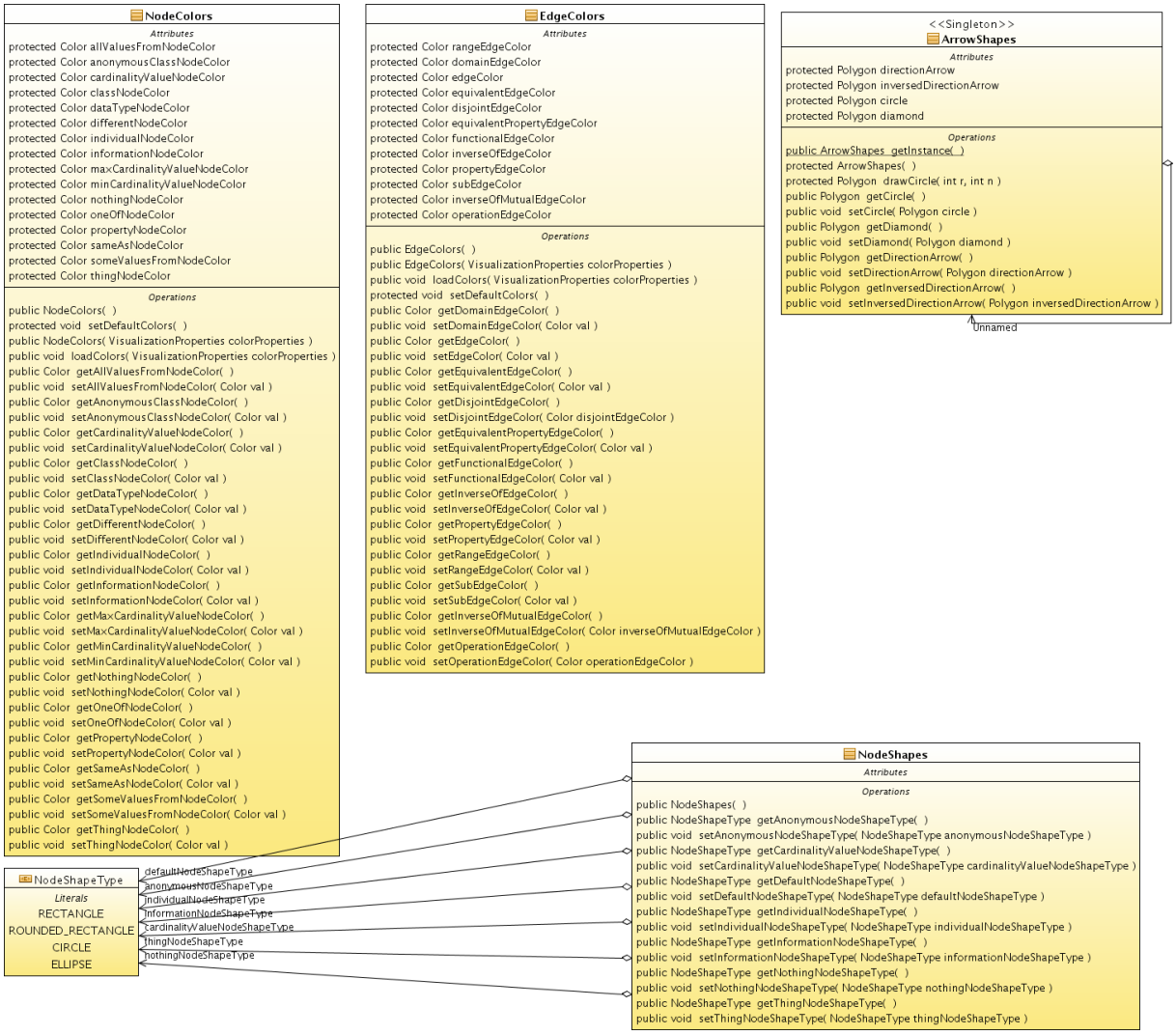
P004	visualization
Opis:	Zawiera dodatkowe klasy przydatne w wizualizacji.
Interfejsy:	
Realizowane wymagania:	WF001, WF008, WI004
Priorytet:	średnio ważne

P005	graph
Opis:	Pakiet zawiera klasy, które zawierają podstawowe operacje na danych OwlApi oraz graph.
Interfejsy:	
Realizowane wymagania:	WD001
Priorytet:	bardzo ważne

P006	utils
Opis:	Pakiet zawiera klasy pomocnicze
Interfejsy:	
Realizowane wymagania:	CF005
Priorytet:	bardzo ważne

5.2 Pakiet options

5.2.1 Diagram



5.2.2 Opis klasy

CO001	EdgeColors
Opis:	Zawiera definicje kolorów dla poszczególnych rodzajów krawędzi.
Klasy nadrzędne:	

Atrybuty:	• domainEdgeColor • edgeColor • equivalentEdgeColor • equivalentPropertyEdgeColor • functionalEdgeColor • inverseOfEdgeColor • propertyEdgeColor • rangeEdgeColor • subEdgeColor
Metody:	
Realizowane wymagania:	WF002
Priorytet:	średnio ważny

CO002	NodeColors
Opis:	Zawiera definicje kolorów dla poszczególnych rodzajów krawędzi.
Klasy nadrzędne:	

Atrybuty:	• allValuesFromNodeColor • cardinalityNodeColor • cardinalityValueNodeColor • classNodeColor • complementOfNodeColor • dataTypeNodeColor • differentNodeColor • functionalPropertyNodeColor • individualNodeColor • informationNodeColor • intersectionOfNodeColor • inverseFunctionalNodeColor • maxCardinalityValueNodeColor • minCardinalityValueNodeColor • nothingNodeColor • oneOfNodeColor • propertyNodeColor • sameAsNodeColor • someValuesFromNodeColor • symmetricPropertNodeColor • thingNodeColor • transitivePropertyNodeColor • unionOfNodeColor
Metody:	
Realizowane wymagania:	WF002
Priorytet:	średnio ważny

CO003	ArrowShapes
Opis:	Singleton przechowujący kształty grotów dla strzałek.
Klasy nadrzędne:	

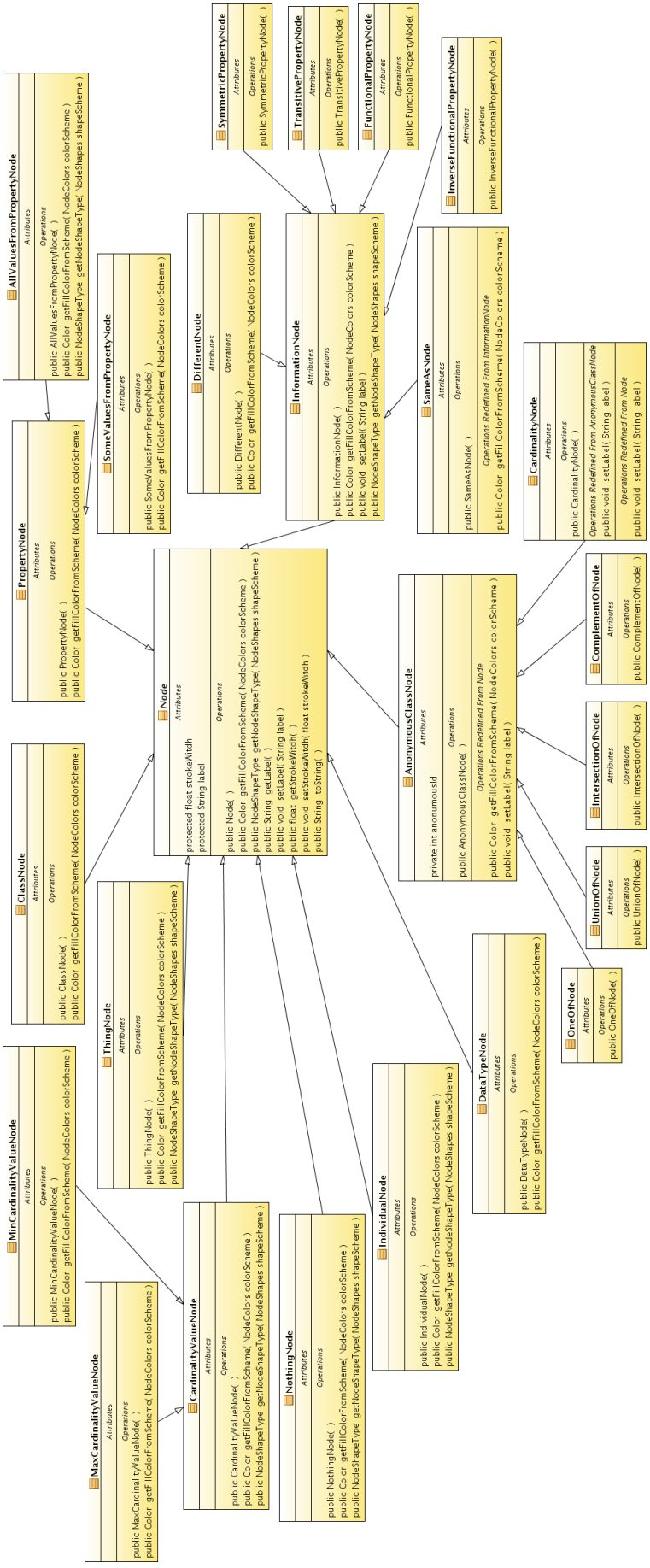
Atrybuty:	• directionArrow • inversedDirectionArrow • circle • diamond
Metody:	
Realizowane wymagania:	WF002
Priorytet:	średnio ważny

CO004	NodeShapes
Opis:	Klasa przechowująca informacje o kształtach poszczególnych węzłów.
Klasy nadrzędne:	
Atrybuty:	• defaultNodeShapeType • anonymousNodeShapeType • individualNodeShapeType • informationNodeShapeType • cardinalityValueNodeShapeType • thingNodeShapeType • NodeShapeType nothingNodeShapeType
Metody:	
Realizowane wymagania:	WF002
Priorytet:	średnio ważny

CO005	NodeShapeType
Opis:	Enum - rodzaje kształtów dla węzłów grafu.
Klasy nadrzędne:	
Atrybuty:	• RECTANGLE • ROUNDED_RECTANGLE • CIRCLE • ELLIPSE
Metody:	
Realizowane wymagania:	WF002
Priorytet:	średnio ważny

5.3 Pakiet nodes

5.3.1 Diagram



5.3.2 Opis klasy

CN001	Node
Opis:	Klasa nadrzędna względem wszystkich klas obsługi wierzchołków. Zawiera definicje podstawowych atrybutów i metod.
Klasy nadrzędne:	
Atrybuty:	• strokeWidth • label
Metody:	• getFillColorFromScheme - zwraca kolor wypełnienia ustawiony dla tego węzła w zadanym schemacie • getNodeShapeType - zwraca kształt węzła
Realizowane wymagania:	WF004, WF005, WF006, WF007, WI004
Priorytet:	bardzo ważne

CN002	AllValuesFromPropertyNode
Opis:	Klasa reprezentuje wierzchołek, będący OWL Property typu AllValuesFrom.
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	• getFillColorFromScheme • getNodeShapeType
Realizowane wymagania:	WF004, WF006, WF007, WI004
Priorytet:	ważne

CN003	AnonymousClassNode
Opis:	Klasa reprezentuje wierzchołek klas anonimowych OWL.
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	• anonymousId
Metody:	• getFillColorFromScheme
Realizowane wymagania:	WF005, WI004
Priorytet:	ważne

CN004	CardinalityNode
Opis:	Klasa reprezentuje wierzchołek klas anonimowych OWL będących wynikiem ograniczenia kardynalności.
Klasy nadrzędne:	CN003 (AnonymousNode)
Atrybuty:	

Metody:	
Realizowane wymagania:	WF007, WI004
Priorytet:	ważne

CN005	CardinalityValueNode
Opis:	Klasa reprezentuje wierzchołek z dokładnym ograniczeniem kardynalności (OWL Cardinality).
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	• getFillColorFromScheme • getNodeShapeType
Realizowane wymagania:	WF007, WI004
Priorytet:	ważne

CN006	ClassNode
Opis:	Klasa reprezentuje wierzchołek OWL Class.
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	• getFillColorFromScheme
Realizowane wymagania:	WF004, WF005, WI004
Priorytet:	ważne

CN007	ComplementOfNode
Opis:	Klasa reprezentuje wierzchołek klas anonimowych OWL będących wynikiem dopełnienia (OWL ComplementOf).
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CN008	DataTypeNode
Opis:	Klasa reprezentuje wierzchołek OWL DataType.
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	• getFillColorFromScheme
Realizowane wymagania:	WF004, WI04
Priorytet:	ważne

CN009	DifferentNode

Opis:	Klasa reprezentuje wierzchołek oznaczający relację DifferentFrom lub AllDifferent pomiędzy wystąpieniami klas (OWL Individual).
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	• getFillColorFromScheme
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CN010	FunctionalPropertyNode
Opis:	Klasa reprezentuje wierzchołek oznaczający, że dane OWL Property to FunctionalProperty.
Klasy nadrzędne:	CN010 (InformationNode)
Atrybuty:	
Metody:	
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CN011	IndividualNode
Opis:	Klasa reprezentuje wierzchołek instancji OWL Individual.
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	• getFillColorFromScheme • getNodeShapeType
Realizowane wymagania:	WF004, WI004
Priorytet:	ważne

CN012	InformationNode
Opis:	Klasa ta jest klasą nadrzędną, dla klas wierzchołków reprezentujących informacje o różnych właściwościach OWL Property.
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	• getFillColorFromScheme • getNodeShapeType
Realizowane wymagania:	WF010, WI004
Priorytet:	ważne

CN013	IntersectionOfNode
Opis:	Klasa reprezentuje wierzchołek klas anonimowych OWL będących wynikiem przecięcia (OWL IntersectionOf).

Klasy nadrzędne:	CN003 (AnonymousNode)
Atrybuty:	
Metody:	
Realizowane wymagania:	WF005, WI004
Priorytet:	ważne

CN014	InverseFunciotnalPropertyNode
Opis:	Klasa reprezentuje wierzchołek oznaczający, że dane OWL Property to InverseFunctionalProperty.
Klasy nadrzędne:	CN010 (InformationNode)
Atrybuty:	
Metody:	
Realizowane wymagania:	WF007, WI004
Priorytet:	ważne

CN015	MaxCardinalityValueNode
Opis:	Klasa reprezentuje wierzchołek ograniczenia kardynalności OWL MaxCardinality.
Klasy nadrzędne:	CN005 (CardinalityValueNode)
Atrybuty:	
Metody:	• getFillColorFromScheme
Realizowane wymagania:	WF007, WI004
Priorytet:	ważne

CN016	MinCardinalityValueNode
Opis:	Klasa reprezentuje wierzchołek ograniczenia kardynalności OWL MinCardinality.
Klasy nadrzędne:	CN005 (CardinalityValueNode)
Atrybuty:	
Metody:	• getFillColorFromScheme
Realizowane wymagania:	WF007, WI004
Priorytet:	ważne

CN017	NothingNode
Opis:	Klasa reprezentuje wierzchołek OWL Nothing.
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	• getFillColorFromScheme • getNodeShapeType
Realizowane wymagania:	WF004, WF005, WI004
Priorytet:	ważne

CN018	OneOfNode
Opis:	Klasa reprezentuje wierzchołek klas anonimowych OWL reprezentujących 1 z klas określonego zbioru (wynik OWL OneOf).
Klasy nadrzędne:	CN003 (AnonymousClassNode)
Atrybuty:	
Metody:	
Realizowane wymagania:	WF005, WF006, WI004
Priorytet:	ważne

CN019	PropertyNode
Opis:	Klasa reprezentuje wierzchołek OWL Property.
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	• getFillColorFromScheme
Realizowane wymagania:	WF004, WF007, WI004
Priorytet:	ważne

CN020	SameAsNode
Opis:	Klasa reprezentuje wierzchołek oznaczający relację OWL SameAs pomiędzy wystąpieniami klas (OWL Individual).
Klasy nadrzędne:	CN010 (InformationNode)
Atrybuty:	
Metody:	• getFillColorFromScheme
Realizowane wymagania:	WF005, WF006, WI004
Priorytet:	ważne

CN021	SomeValuesFromPropertyNode
Opis:	Klasa reprezentuje wierzchołek, będący OWL Property typu SomeValuesFrom.
Klasy nadrzędne:	CN019 (PropertyNode)
Atrybuty:	
Metody:	• getFillColorFromScheme
Realizowane wymagania:	WF005, WF006, WI004
Priorytet:	ważne

CN022	SymmetricPropertNode
Opis:	Klasa reprezentuje wierzchołek oznaczający, że dane OWL Property to SymmetricProperty.
Klasy nadrzędne:	CN010 (InformationNode)
Atrybuty:	
Metody:	

Realizowane wymagania:	WF007, WI004
Priorytet:	ważne

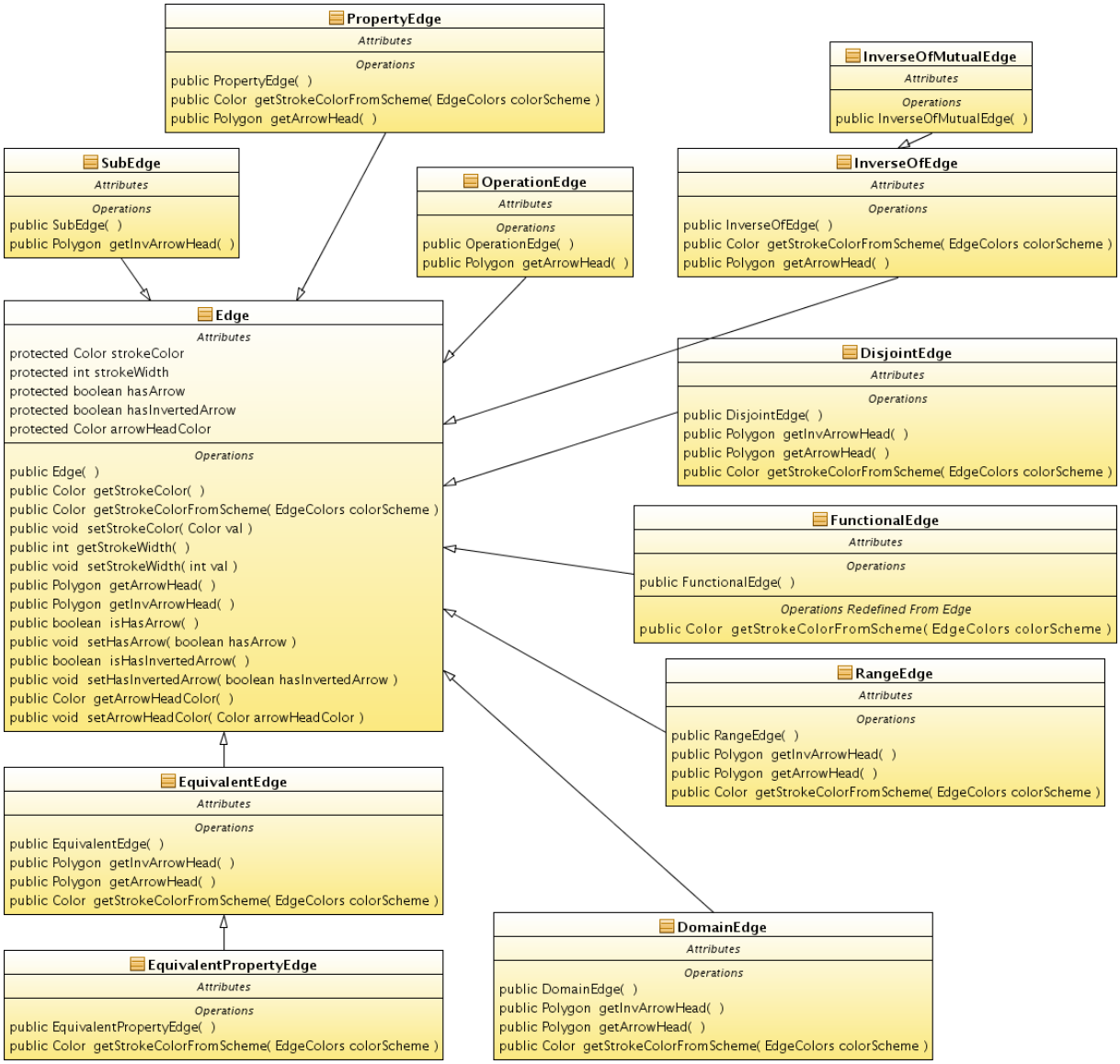
CN023	ThingNode
Opis:	Klasa reprezentuje wierzchołek OWL Thing.
Klasy nadrzędne:	CN001 (Node)
Atrybuty:	
Metody:	• getFillColorFromScheme • getNodeShapeType
Realizowane wymagania:	WF004, WF005, WI004
Priorytet:	ważne

CN024	TreansitivePropertyNode
Opis:	Klasa reprezentuje wierzchołek oznaczający, że dane OWL Property to TransitiveProperty.
Klasy nadrzędne:	CN010 (InformationNode)
Atrybuty:	
Metody:	
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CN025	UnionOfNode
Opis:	Klasa reprezentuje wierzchołek klas anonimowych OWL będących wynikiem unii (OWL UnionOf).
Klasy nadrzędne:	CN003 (AnonymousNode)
Atrybuty:	
Metody:	
Realizowane wymagania:	WF005, WF006, WI004
Priorytet:	ważne

5.4 Pakiet edges

5.4.1 Diagram



5.4.2 Opis klasy

CE001	Edge
Opis:	Klasa reprezentująca prostą krawędź na grafie. Jest nadklasą dla pozostałych klas krawędzi.
Klasy nadrzędne:	
Atrybuty:	• boolean hasArrow • boolean hasInvertedArrow • Color arrowHeadColor • Color strokeColor • int strokeWidth

Metody:	• getStrokeColorFromScheme • getArrowHead • getInvArrowHead
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	bardzo ważne

CE002	DisjointEdge
Opis:	Klasa reprezentująca krawędź oznaczającą rozłączność klas (OWL Disjoint).
Klasy nadrzędne:	CE001 (Edge)
Atrybuty:	
Metody:	• getStrokeColorFromScheme • getArrowHead • getInvArrowHead
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CE003	DomainEdge
Opis:	Klasa reprezentująca krawędź łączącą Property z klasą właściwości OWL DomainOf.
Klasy nadrzędne:	CE001 (Edge)
Atrybuty:	
Metody:	• getStrokeColorFromScheme • getArrowHead • getInvArrowHead
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CE004	EquivalentEdge
Opis:	Klasa reprezentująca krawędź oznaczającą równoznaczność (OWL Equivalent).
Klasy nadrzędne:	CE001 (Edge)
Atrybuty:	
Metody:	• getStrokeColorFromScheme • getArrowHead • getInvArrowHead

Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CE005	EquivalentPropertyEdge
Opis:	Klasa reprezentująca krawędź oznaczającą równoznaczność OWL Property (OWL EquivalentProperty).
Klasy nadrzędne:	CE004 (EquivalentEdge)
Atrybuty:	
Metody:	• getStrokeColorFromScheme
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CE006	FunctionaltEdge
Opis:	Klasa reprezentująca krawędź łączącą wierzchołki InformationNode(CN012) z OWL Property, którego dotyczy.
Klasy nadrzędne:	CE001 (Edge)
Atrybuty:	
Metody:	• getStrokeColorFromScheme
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CE007	InverseOfEdge
Opis:	Klasa reprezentująca krawędź oznaczającą odwrotność (OWL InverseOf).
Klasy nadrzędne:	CE001 (Edge)
Atrybuty:	
Metody:	• getStrokeColorFromScheme • getArrowHead • getInvArrowHead
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CE008	PropertyEdge
Opis:	Klasa reprezentująca krawędź oznaczającą relację między Property a klasą.
Klasy nadrzędne:	CE001 (Edge)
Atrybuty:	

Metody:	• getStrokeColorFromScheme • getArrowHead • getInvArrowHead
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CE009	RangeEdge
Opis:	Klasa reprezentująca na grafie krawędź łączącą Property z klasą właściwości OWL Range.
Klasy nadrzędne:	CE001 (Edge)
Atrybuty:	
Metody:	• getStrokeColorFromScheme • getArrowHead • getInvArrowHead
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CE010	SubEdge
Opis:	Klasa reprezentująca krawędź związku OWL SubClass pomiędzy klasami.
Klasy nadrzędne:	CE001 (Edge)
Atrybuty:	
Metody:	• getInvArrowHead
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

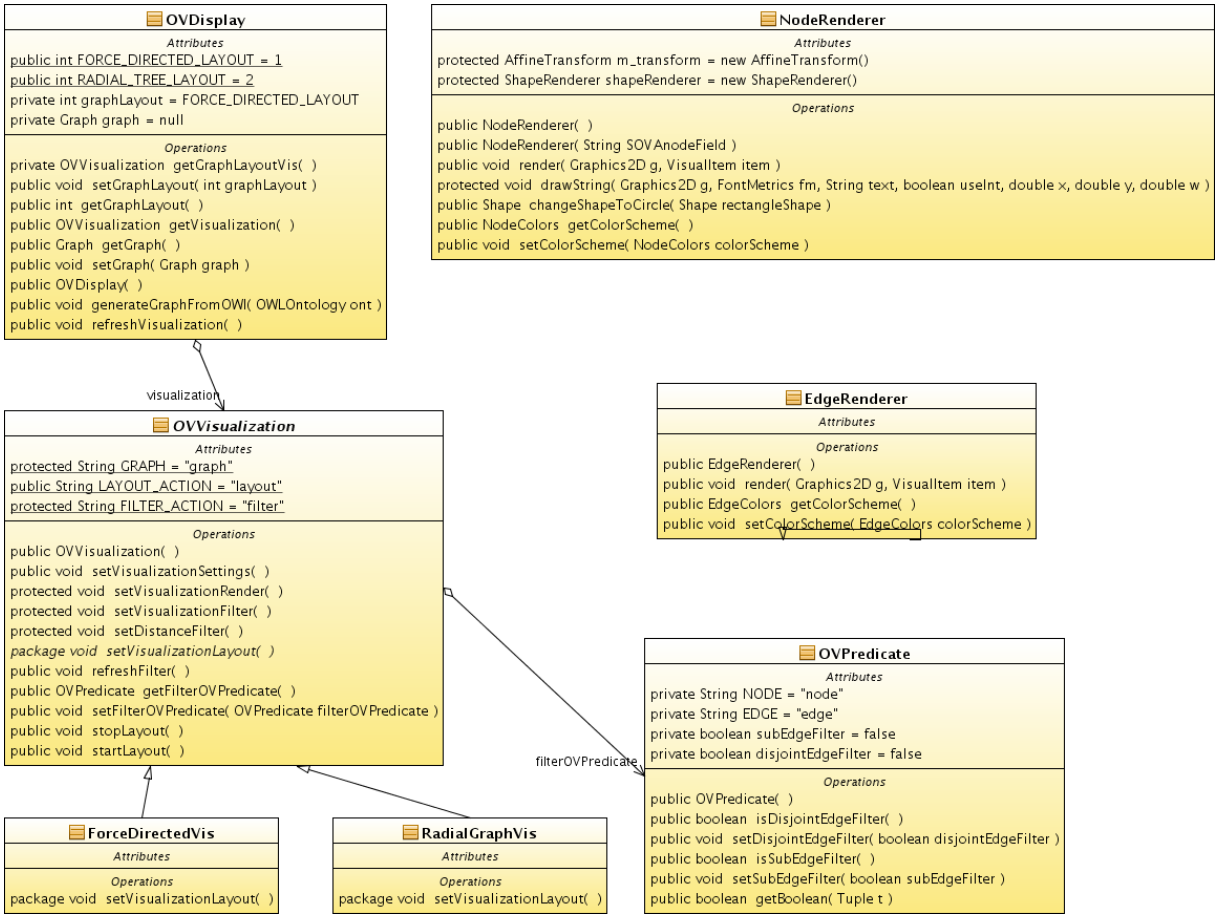
CE011	InverseOfMutualEdge
Opis:	Klasa reprezentująca krawędź oznaczającą wzajemną odwrotność (OWL InverseOf) property.
Klasy nadrzędne:	CE007 (InverseOfEdge)
Atrybuty:	
Metody:	
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

CE012	OperationEdge
Opis:	Krawędź do oznaczania powiązań operacji, w wyniku których powstają klasy anonimowe.
Klasy nadrzędne:	CE001 (Edge)
Atrybuty:	

Metody:	• getArrowHead
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	ważne

5.5 Pakiet visualization

5.5.1 Diagram



5.5.2 Opis klasy

CV001	EdgeRenderer
Opis:	Klasa przeciążająca metody renderowania krawędzi grafu z biblioteki prefuse.
Klasy nadrzędne:	prefuse.render.EdgeRenderer
Atrybuty:	
Metody:	• render(Graphics2D g, VisualItem item) - metoda renderująca krawędź
Realizowane wymagania:	WF001, WF008, WI004
Priorytet:	ważne

CV002	NodeRenderer
Opis:	Klasa przeciążająca metody renderowania wierzchołków grafu z biblioteki prefuse.
Klasy nadrzędne:	prefuse.render.LabelRenderer
Atrybuty:	
Metody:	• drawString(Graphics2D g, FontMetrics fm, String text, boolean useInt, double x, double y, double w) - metoda wypisujaca na wierzchołku String • render (Graphics2D g, VisualItem item) - metoda renderująca wierzchołek • Shape changeShapeToCircle(Shape rectangleShape) - Zamienia domyślny, prostokątny kształt węzła na koło. Koło to ma średnicę równą szerokości lub wysokości węzła (większa z wartości) i środek w tym samym punkcie.
Realizowane wymagania:	WF001, WF008, WI004
Priorytet:	ważne

CV003	OVDisplay
Opis:	Klasa tworząca obiekt JComponent do umieszczenia na okienku JAVA zawierający wygenerowany graf z wizualizacją
Klasy nadrzędne:	prefuse.Display
Atrybuty:	• Graph graph - obiekt typu prefuse.data.graph zawierajacy dane o grafie do wyświetlenia.
Metody:	• generateGraphFromOWl(OWLOntology ont) - wpisuje do klasy obiekt Grpah wygenrowany na podstawie ontologii • refreshVisualization()
Realizowane wymagania:	WF001, WF002, WF008, WI004
Priorytet:	ważne

CV005	ForceDirectedVis
Opis:	Klasa wizualizujące grafy w oparciu o algorytm ForceDirected
Klasy nadrzędne:	CV007 (OVVisualization)
Atrybuty:	
Metody:	
Realizowane wymagania:	WF001, WF002, WF008, WI004
Priorytet:	ważne

CV006	OVPredicate

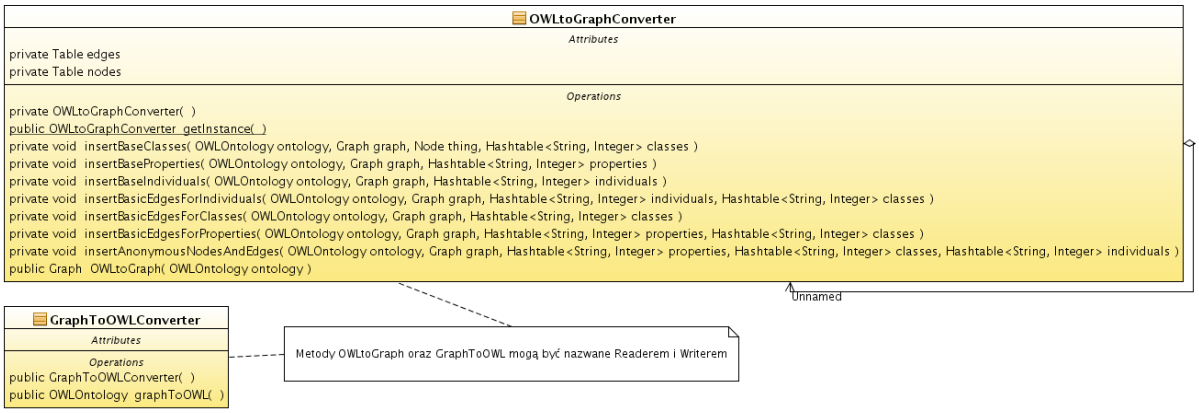
Opis:	Klasa zawierająca predykaty filtrowania obiektów w wyswietlanych grafach
Klasy nadrzędne:	prefuse.data.expression.AbstractPredicate
Atrybuty:	• boolean disjointEdgeFilter • boolean subEdgeFilter
Metody:	
Realizowane wymagania:	WF001, WF002, WF008, WI004
Priorytet:	ważne

CV007	OVVisualization
Opis:	Klasa obsługi wizualizacji.
Klasy nadrzędne:	prefuse.Visualization
Atrybuty:	
Metody:	• refreshFilter() • startLayout() • stopLayout()
Realizowane wymagania:	WF001, WF002, WF008, WI004
Priorytet:	ważne

CV008	RadialGraphVis
Opis:	Klasa wizualizująca graf w oparciu o algorytm RadialGraph
Klasy nadrzędne:	CV007 (OVVisualization)
Atrybuty:	
Metody:	
Realizowane wymagania:	WF001, WF002, WF008, WI004
Priorytet:	średnio ważne

5.6 Pakiet graph

5.6.1 Diagram



5.6.2 Opis klasy

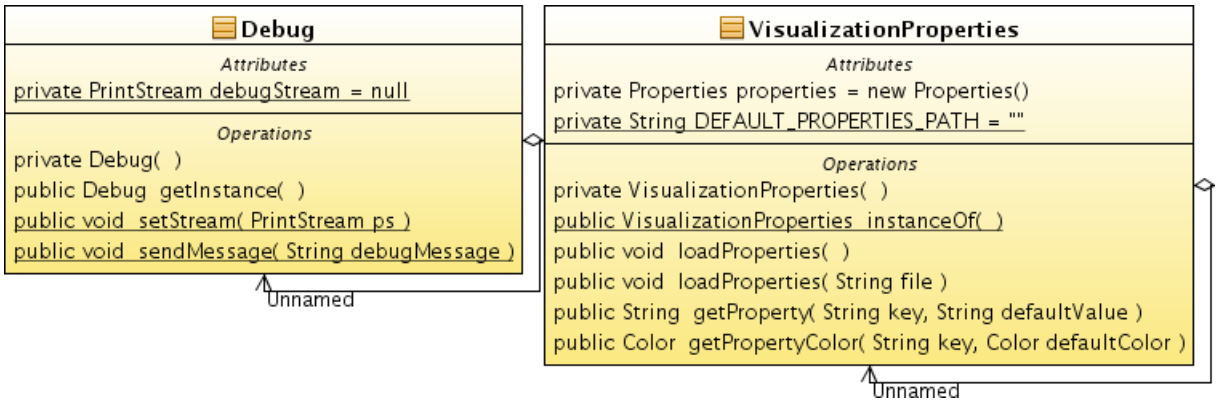
CG001	GraphToOWLConverter
Opis:	Klasa zawierająca metody pozwalające na przetwarzanie obiektów grafów z prefuse na obiekty OWL API. Klasa jest singletonem.
Klasy nadrzędne:	
Atrybuty:	• INSTANCE - instancja klasy GraphToOWLConverter
Metody:	• getInstance() - zwraca instancję klasy • GraphToOWL(OWLOntology ontology) -Zamienia graf z biblioteki prefuse na ontologię zapisana w OWL API.
Realizowane wymagania:	WD001, WI004
Priorytet:	ważne

CG002	OWLtoGraphConverter
Opis:	Klasa zawierająca metody pozwalające na przetwarzanie obiektów OWL API na obiekty prefuse. Klasa jest singletonem.
Klasy nadrzędne:	
Atrybuty:	• INSTANCE - instancja klasy GraphToOWLConverter • Table edges • Table nodes

Metody:	• getInstance() - zwraca instancję klasy • insertAnonymousNodesAndEdges(OWLOntology ontology, Graph graph, Hashtable properties, Hashtable classes, Hashtable individuals) • insertBaseClasses(OWLOntology ontology, Graph graph, Node thing, Hashtable classes) • insertBaseProperties(OWLOntology ontology, Graph graph, Hashtable properties) • insertBaseIndividuals(OWLOntology ontology, Graph graph, Hashtable individuals) • insertBasicEdgesForIndividuals(OWLOntology ontology, Graph graph, Hashtable individuals,Hashtable classes) • insertBasicEdgesForClasses(OWLOntology ontology, Graph graph, Hashtable classes) • insertBasicEdgesForProperties(OWLOntology ontology, Graph graph, Hashtable properties, Hashtable classes) • OWLToGraph(OWLOntology ontology) -Zamienia ontologię w OWL API na graf z biblioteki prefuse.
Realizowane wymagania:	WD001, WI004
Priorytet:	ważne

5.7 Pakiet utils

5.7.1 Diagram



5.7.2 Opis klasy

CU001	Debug
Opis:	Klasa do użycia przy debugowaniu, zapewnia strumien z błędami zwracanymi przez bibliotekę. Klasa jest singletonem.
Klasy nadrzędne:	
Atrybuty:	• INSTANCE - instacja klasy Debug • debugStream - Strumień do którego wpisywane są informacje potrzebne do debugowania
Metody:	• getInstance() - zwraca instację klasy • setStream(PrintStream ps) - ustawia podany strumień jako strumień na który zwracane będa błędy • sendMessage(String s) - wysyła wiadomość na strumień do debugowania, jeżeli został wcześniej podpięty za pomocą funkcji setStream
Realizowane wymagania:	WF006, WF007, WI004
Priorytet:	bardzo ważne

CU002	VisualizationProperties
Opis:	Klasa odpowiada za wczytywanie ustawień kolorów dla węzłów oraz krawędzi z wybranego lub domyślnego.
Klasy nadrzędne:	
Atrybuty:	• properties
Metody:	• loadProperties
Realizowane wymagania:	WF002
Priorytet:	ważne

6 Słownik

Symbol projektu: 3@KASK	Opiekun projektu: mgr inż. Tomasz Boiński
Nazwa Projektu: Wizualizacja grafów za pomocą biblioteki Prefuse	

Nazwa Dokumentu: Słownik pojęć	Nr wersji: 0.04
Odpowiedzialny za dokument: Piotr Orłowski	Data pierwszego sporządzenia: 31.03.09
Przeznaczenie: WEWNĘTRZNE	Data ostatniej aktualizacji: 15.05.09

Historia dokumentu

Wersja	Opis modyfikacji	Rozdział/strona	Autor modyfikacji	Data
1	Stworzenie zarysu słownika	wszystkie	Anna Jaworska	31.03.09
2	Podstawowe pojęcia Semantic Web	Pojęcia ogólne	Piotr Orłowski	31.03.09
3	Licencje wolnego oprogramwania	Pojęcia ogólne	Piotr Orłowski	07.04.09
4	Uzupełnienie brakujących pojęć	wszystkie	Piotr Orłowski	15.06.09

6.1 Jak korzystać ze slownika

Słownik został podzielony na dwie części:

- pojęcia ogólne
- pojęcia specyficzne dla projektu.

Pojęcia zostały podane w sposób alfabetyczny. Słownik ten będzie rozwijany na bieżąco razem z rozwijaniem całego projektu.

6.2 Pojęcia ogólne

agent (lm. agenty) jednostka (np. program), działającą w pewnym środowisku, zdolna do komunikowania się, monitorowania swego otoczenia i podejmowania autonomicznych decyzji, aby osiągnąć cele określone podczas jej projektowania lub działania.

API ang. Application Programming Interface, interfejs dla programów, zestaw poleceń, funkcji, metod, formatów i danych, które służą do wymiany informacji pomiędzy aplikacją i systemem operacyjnym oraz innymi programami lub sterownikami.

aplikacja standalone to aplikacja, która do uruchomienia nie wymaga innych programów

BSD Berkeley Software Distribution License, jedna z licencji zgodnych z zasadami Wolnego Oprogramowania stworzona na Uniwersytecie Kalifornijskim w Berkeley.

debugowanie znany także jako odpluskwianie, proces szukania i naprawiania błędów w programach komputerowych za pomocą specjalnych narzędzi do tego przeznaczonych.

GPL GNU General Public License, jedna z licencji Wolnego Oprogramowania stworzona przez Richarda Stallmana i Ebena Moglena; zawiera zastrzeżenie, że wszystkie pochodne prace bazujące na kodzie wydanym na licencji GPL muszą być wydane na licencji GPL.

JAVA Obiektowy język programowania; pojęcie używane czasem w sensie maszyny wirtualnej jezyka JAVA

javadoc - generator dokumentacji stworzony przez firmę Sun Microsystems; narzędzie to generuje dokumentację kodu źródłowego Javy na podstawie zamieszczonych w kodzie komentarzy javadoc(do ich tworzenia używa się specjalnych tagów, które pozwalają na prawidłową interpretację informacji tam zawartej).

JVM - Java Virtual Machine, maszyna wirtualna Javy, niezależny od platformy system uruchomieniowy dla programów napisanych w języku Java oraz innych (np. Jython) językach.

kapsułkowanie - znane również jako hermetyzacja, enkapsulacja (z ang. encapsulation), jedno z założeń paradygmatu programowania obiektowego. Polega ono na ukrywaniu pewnych danych składowych lub metod obiektów danej klasy tak, aby były one dostępne tylko metodom wewnętrznym danej klasy oraz, ewentualnie, wybranym innym obiektom (np. klas zaprzyjaźnionych)..

KASK Katedra Architektury Systemów Komputerowych WETI

krotka - pojęcie matematyczne oznaczające uporządkowany, skończony zbiór elementów; w informatyce często używane do określenia rekordu bazy danych. W przypadku prefuse odnosi się do pojedynczego rekordu w tabeli.

metadane są to dane opisujące inne dane, stosowane w celu ułatwienia korzystania z tych danych.

OCS Ontology Creation System - projekt realizowany w ramach grantu (tu id grantu) na KASK-u.

ontologia dział filozofii starający się badać strukturę rzeczywistości i zajmujący się problematyką związaną z pojęciami bytu, istoty, istnienia i jego sposobów, przedmiotu i jego własności, przyczynowości, czasu, przestrzeni, konieczności i możliwości.

OWL Web Ontology Language, jest to rozszerzenie RDFS. Język do opisu ontologii stworzony przez W3C.

pakiet - tutaj jednostka organizacji klas w programowaniu obiektowym.

Prefuse Biblioteka języka JAVA, pozwalająca na estetyczna prezentacje danych, w szczególności grafów

RDF Resource Description Framework, jest specyfikacją W3C stosowaną do modelowania metadanych w postaci wyrażeń zawierających predykaty, klasy i podmioty; wyrażenia te tworzą graf skierowany.

RDFS RDF Schema, język reprezentacji wiedzy oparty na RDF.

Sieć Semantyczna ang. Semantic Web, projekt, który ma umożliwić łatwiejsze i bardziej logiczne wyszukiwanie przez maszyny i programy(agenty) danych w sieci Internet; znaczenie zasobów informacyjnych opisywane jest tu przy pomocy ontologii; do standardów rozwijanych wraz z Semantic Web należą m.in. OWL, RDF oraz RDFS

SHOIN/OWL - język do wyrażania logiki opisowej ontologii.

strumień błędów - specjalny strumień danych w programie, na który kierowane są informacje o błędach oraz ewentualnie przebiegu działania funkcji programu, w których istnieje ryzyko wystąpienia błędów.

SVN SubVersioN - system kontroli wersji.

W3C World Wide Web Consorcium - organizacja odpowiedzialna za ustalanie standardów dla metajęzyków.

WETI/ETI Wydział Elektroniki, Telekomunikacji i Informatyki Politechniki Gdańskiej

XML ang. Extensible Markup Language, uniwersalny język formalny przeznaczony do reprezentowania różnych danych w ustrukturalizowany sposób.

6.3 Pojęcia specificzne dla projektu

kardynalność tutaj występująca w języku OWL liczność elementu

klasa anonimowa tutaj klasa będąca wynikiem operacji (np. logicznej) na innych klasach bądź powstała przez wyliczenie instancji.

portalSubsystem część projektu OCS, pozwala na wizualizację online plików OWL

7 Załączniki

1. Notatka1
2. Notatka2
3. Notatka3
4. Notatka4
5. Notatka5
6. Notatka6
7. Notatka7
8. Notatka8
9. Notatka9
10. Notatka10
11. Notatka11
12. Notatka12
13. Notatka13
14. Notatka14
15. Notatka15
16. Notatka Specjalna
17. Plakat